CHAPTER 2

A Programmer's Overview of Windows 2000

In This Chapter

Learn about the new features in Windows 2000

Understand the various low-level services offered by Windows 2000

Discover how distributed services can help you in today's corporate environment

Determine how to keep your network safe with new security features

Get an overview of the latest in COM

By now, you've probably heard just about everything you need to know about Windows 2000 from a user's perspective. Microsoft has told you that Windows 2000 is going to be more reliable, faster, require less administrative time, and still provide phenomenal new features. The problem is that you're a programmer, and though these user features are nice, you've got an application to build. This chapter isn't going to answer all of your questions; we have a whole book in which to do that. What this chapter *will* do is give you an overview of what Windows 2000 has to offer. It'll tantalize your taste buds and prepare you for the information to follow.

Just as the user will get a list of new features in their documentation, we'll cover a list of the new features for the programmer in the first section of this chapter. I'm not going to provide a detailed explanation of how each new feature works, but I'll give you a basic description of what's going on. By the time you finish this first section, you should at least know which new features fit into your next application.

The next section will help you understand the new low-level features that Windows 2000 has to offer. We'll group these new features into the various core groups that you're used to working with. You'll be surprised at just how many new features that Windows 2000 has to offer. This section might help you understand why it took Microsoft so long to get this product out the door.

The LAN has given way to something bigger. Networks no longer reside in the same building, the same state, the same country, or, in the case of NASA, the same planet. It's not difficult to figure out that a way is needed to keep all of the disparate parts of the network communicating with each other. We'll look at how two new Windows 2000 features, Active Directory and Asynchronous Transfer Mode (ATM), can help you from a programming perspective. We'll spend quite a bit of time looking at Active Directory in Chapters 19 (theory) and 20 (application).

Security, as always, is a big issue that every programmer needs to deal with at some level in the applications he or she creates. Windows 2000 offers some new security features that make keeping your network safe a little easier. The fourth section of the chapter is going to provide you with a brief overview of this topic. Chapters 17 (theory) and 18 (application) will help you understand these new features in much greater detail.

Finally, we'll take a look at COM. It seems like Microsoft is constantly tinkering with their component technology, and Windows 2000 is no different. In this case, they've even given the technology changes a new name, COM+. We'll be visiting COM+ throughout the book because it affects just about every area of Windows 2000 programming. Be sure you take the time to look at the overview in this chapter before you delve into the more detailed information that follows in other chapters.

The information contained in this chapter was current as of the time of writing. It's based on both Microsoft documentation and the Platform SDK. However, Microsoft has changed their product releases right up to the point of release to manufacturing in the past, and I don't expect things to be any different this time. In addition, there are usually some documentation differences due to a variety of geo-political needs such as differences in language support and export restrictions. As a result, some of the information you find in this chapter may not exactly reflect the state of the released product. Make sure you check the Platform SDK and current Microsoft documentation to get the latest information about Windows 2000.

New Features for Windows 2000

In this section of the chapter we are going to look at both new Windows 2000 features and those that have improved from Windows NT. For the most part, this section avoids user-only features — it'll concentrate instead on the programmer view of things. The following section will tell you what's new to Windows 2000. Some of these features, like Windows Driver Model (WDM), have already appeared in other versions of Windows, like Windows 98. After we've covered all of the new features, we'll look at Windows 2000's improvements over Windows NT 4.0.

It's important for you to know what I mean when I say "new." For the purposes of this chapter, new is anything that didn't appear in the original version of Windows NT 4.0 without any service packs installed. It seems like Microsoft always adds some new feature to their service packs, which makes the term "new" a moving target.

What's New

Windows 2000 will include a lot of new features that programmers are going to love and users will probably barely notice. One of the features that fall into this category is the Windows Driver Model (WDM). No longer will you have to write two versions of every driver that you create — all new versions of Windows will share the same drivers. Not only will this make it faster for you to create a driver, but the user will appreciate the fact that you can get the drivers done faster and that they'll actually work on all the user's machines.

Even though the user will notice the new power-management features provided by Windows 2000, they won't appreciate everything that Microsoft has put under the hood of this new operating system. While most of us don't spend a lot of time working with power management, there are times when you might want to know when the user's battery is about to run out of power. The new power-management API features will make this a lot easier and also more accurate.

The file system is totally new for Windows 2000 as well. Yes, Microsoft still officially classifies it as NTFS, but the version number has gone up to 5 to reflect changes in capability. For one thing, NTFS 5 includes support for Active Directory, which is something you won't find in Windows NT 4.0. (We'll discuss Active Directory in Chapters 19 and 20.) Windows NT 4.0 uses NTFS 4. Needless to say, the two versions of NTFS aren't compatible, so you can't really mix them without help. You must install Service Pack 4 on any Windows NT 4.0 machine that you want to use to access drives containing NTFS 5. There is one big change the user might notice: NTFS 5 provides support for FAT32, which means that you can use the more efficient FAT32 file system for dual-boot machines. FAT 32 is the 32-bit version of the file allocation table (FAT) disk format provided with the OSR2 version of Windows 95 and all versions of Windows 98. NTFS 5 also supports the Universal Disk Format (UDF), which is based on ISO 13346. This file system replaces the older Compact Disk File System (CDFS), which is based on ISO 9660. You must have UDF to support new disk technologies like digital video (or versatile) disk (DVD). There are a lot of other NTFS 5 changes, as well, such as reparse points, which we'll discuss as the book progresses.

The version of UDF used in both Windows 2000 (UDF 1.50) and Windows 98 (UDF 1.02) is read-only. At the time of this writing, the latest version of UDF is 2.0, and it wouldn't surprise me if this update was found in both Windows 2000 and Windows 98 sometime in the near future. You can find out more about this standard at the Optical Storage Technologies Association Web site at: `http://www.osta.org/`. This site provides a lot of very interesting information about optical storage. For example, there's a paper available on multi-read technologies.

Programmers have user-interface needs, and Microsoft hasn't forgotten that. This version of Windows 2000 provides multi-monitor support. Not only will multiple monitors give you more screen space to use for applications (when the user of your application also has more than one monitor), but now you'll have additional screen space for your own needs. For example, you'll be able to debug in one window and keep an eye on the application in the other.

Trying to administer the vast resources of many networks today is a complex task, one that many administrators are starting to ask programmers to help with in a big way. That's why the new Zero Administration Windows (ZAW) features in Windows 2000 are so important. ZAW isn't just a nice technology that Microsoft is adding to Windows to help the administrator—it's also an API that'll help you write smarter applications that actually respond to both user and administrator needs.

Windows 2000 security is a mix of something old and something new. Kerberos is new and is the replacement for Microsoft's aging Windows NT LAN Managers (NTLM) authentication methodology. You'll also find both Public Key Infrastructure (PKI) and Distributed Password Authentication in this version of Windows. Even though the CryptoAPI has been around for a while, Microsoft has added a little over 100 additional cryptographic and certificate-based functions to this API. We'll discuss security at length later in Chapters 17 and 18 and you can be sure of finding all of these technologies in the discussion.

What's Improved

Now that we've covered the new part of the picture, let's look at what is improved. Given Microsoft's emphasis on the Internet these days, it's not too surprising that you'll find some new Internet features in Windows 2000 as well. Even though it has essentially disappeared from sight, Active Desktop got some new features, as did channels. The NetMeeting API has been beefed up as well. You'll find it easier to access various parts of this product, especially when it comes to working with users.

Plug and play is an area in which the user really doesn't notice the underlying technology, just the effect if it doesn't work properly. Some users have called plug and play, plug and pray. They're right—up until now conflicting standards and problem hardware have kept plug and play from working properly. Windows 2000 makes improvements in this area, though I won't go so far as to say that you'll find fixes for everything. However, there is a downside to this new level of support. Microsoft has decided to kill support for hardware that's older than three years, in many cases to ensure that new hardware will work properly. What this means to you as a user is that you might have an unexpected hardware upgrade in your future if you use Windows 2000.

Want to dazzle the user the next time they start your application? DirectX 6.0 provides better support for just about every multimedia element that you can imagine. Not only does DirectX 6.0 support more hardware, but it provides fuller support for existing features as well. For example, you'll find that you gain more in the graphics department. 3D rendering, and special effects such as fog are faster than before. In short, many people are saying that Microsoft is finally getting support for gamers and other heavy multimedia users, right.

You'll also find a lot of additions to the services provided by Windows 2000. In fact, there are so many service additions that they really require their own sections. We'll discuss most of the new service changes that you'll see in Windows 2000 in the sections of the chapter that follow this one. It's also important to be aware of the fact that programmers will have a lot more to choose from when it comes to services in Windows 2000 — perhaps too many choices.

Low-level Services

Windows 2000 is the largest version of Windows yet. You'll find that it provides you with more than enough new services to create applications that use all of the new features of today's hardware. In fact, it's unlikely that any one programmer will even scratch the surface of all of the new features that Windows 2000 has to offer. The following sections will provide you with an overview of the new low-level services that you'll find in this product release.

USER

As you know, the USER subsystem provides a lot of interface elements and defines how input devices work. There are more changes in the USER subsystem than you might initially suspect. In this section, I'll briefly talk about some of the more important USER subsystem features.

One of the first areas that we'll look at in the USER subsystem are some new messages that Windows 2000 supports. While it's true that some of these messages have been around in an unofficial form for some time now, the fact that they're documented means that you can actually use them for application development without concern. Table 2-1 contains a list of the new messages and describes how you can use them.

Table 2-1
New USER Subsystem Messages

Message	*Where Used*	*Description*
WM_GETOBJECT	MSAA	Used by the OLEACC.DLL to retrieve an Active Accessibility server application object. In other words, if you create an MSAA application, you'll need to handle this message in some cases.
WM_MENUCOMMAND	Menu	Sent when the user makes a menu selection.
WM_MENUDRAG	Drag & Drop Menu	Sent when the user begins a drag operating on a drag and drop menu. Input parameters include the affected menu item and its current position.
WM_MENUGETOBJECT	Drag & Drop Menu	Sent when the cursor moves into a drag and drop menu. This message also gets sent if the cursor moves from the middle of the menu item to either the top or the bottom of the menu item.
WM_MENURBUTTONUP	Menu	Sent when the user releases the right mouse button when the mouse cursor is on a menu item.
WM_MOUSEHOVER	Window	Posted when the mouse cursor hangs over the client area of a window for a specific amount of time. You specify the hover time interval using the TrackMouseEvent()function.
WM_MOUSELEAVE	Window	Posted when the mouse cursor leaves the client area of a window. You activate this message using the TrackMouseEvent() function.
WM_MOUSEWHEEL	Window	Sent to the window that currently has focus when the mouse wheel is turned. This message includes the mouse X position, Y position, Z position, and any required control key flags.
WM_NCMOUSEHOVER	Window	Posted when the mouse cursor hangs over the non-client area of a window for a specific amount of time. You specify the hover time interval using the TrackMouseEvent() function.
WM_NCMOUSELEAVE	Window	Posted when the mouse cursor leaves the non-client area of a window. You activate this message using the TrackMouseEvent() function.
WM_UNINITMENUPOPUP	Drop-down Menu	Signals that either a drop-down or submenu has been destroyed.

Speaking of messaging, there's always been an InSendMessage( ) function that tells you whether a message was sent to or posted to your function. Windows 2000 has a new form of this function that tells you a lot more about the calling function. Instead of the simple true/false result you would get in the past, you can now determine whether the calling function is still blocked and whether it used the SendMessageCallback( ), SendNotifyMessage( ), SendMessage( ), or SendMessageTimeout( ) functions to call your function.

Windows NT (and 2000) has always provided the Task Manager to help you monitor the tasks currently being executed on your machine. (You access this feature by right-clicking the Taskbar and choosing Task Manager from the context menu.) In the past, the Processes tab of the Windows Task Manager dialog was pretty limited. You could find out which processes were running and their Process Identifier (PID), but that's about it. Now you can use View | Select Columns command to display a Select Columns dialog that allows you to chose additional information to display on the Processes tab. Figure 2-1 shows what the Set Columns tab looks like. Notice that two possible selections are the USER Objects and GDI Objects that the process is using.

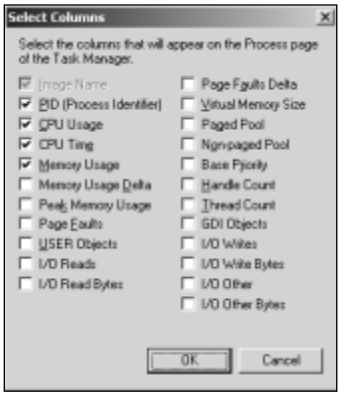

Figure 2-1: The Processes tab of the Task Manager is now much more functional.

One of the new features that affects the USER subsystem actually comes from Microsoft Active Accessibility (MSAA). Up until now, if you wanted to create something like a macro that input keystrokes or mouse movement into the input stream, you had to use one of the KBD_EVENT or MOUSE_EVENT functions. The problem was that these functions only allowed you to input one event at a time, which meant that the user could interrupt the input stream with events of their own. The result was often a humorous trashing of the screen and a failure of the macro. The SendInput API changes all of this. You can now create a sequence of mouse or keyboard events and send them to the input stream with a guarantee that they will all arrive together. This means that macro writing should take a turn for the better in the near future.

Both the WinNotifyEvent API and the SendInput API are part of the larger MSAA package. Unfortunately, like many of the new Windows 2000 features we'll talk about in this chapter, you won't find support for these features in the current Visual C++ 6.0 package (including both Professional and Enterprise Editions). What you'll need, in this case, is a special MSAA SDK that's available at `http://www.microsoft.com/enable/news/msaa1198.htm`.

Tracking the mouse is an important part of many applications. Windows 2000 provides a new function called GetMouseMovePoints( ). You can supply this function with an *x*,*y* coordinate and optional time stamp as input, and it'll return the points to which the mouse has moved since it was at that point. The GetMouseMovePoints( ) function will track the last 64 places the mouse moved, which should be enough for most graphics applications. There are two forms of output from this function: display driver pixel units or original mouse format.

Writing a screen saver? You can use the new GetLastInputInfo( ) function to determine the last time that the system has received any input. This function will return a LASTINPUTINFO data structure that contains a dwTime member that gives the last input time. dwTime is actually in ticks, rather than minutes and seconds, so you'll need to do some conversion if you require the actual time since the last input.

There are a couple of gizmo-type items that you may or may not like. The first is active window tracking. The idea is fairly simple: Moving your cursor to a non-foreground window makes that window the foreground window. In other words, you won't have to click on background windows to activate them anymore. This feature is controlled through the SystemParametersInfo( ) function. The other gizmo feature is window animation. This little processor cycle waster will give your window a special effect like a fade when you initially open, minimize, or maximize it. Besides fading the window, you can also slide or roll it.

For those of you who, in the past, have been frustrated trying to get information out of Windows about your windows, there's a series of new functions designed to make getting information easier: GetWindowInfo( ), GetTitleBarInfo, GetMenu BarInfo, GetScrollBarInfo( ), GetComboBoxInfo( ), GetListBoxInfo( ), and GetWindow FileModuleName( ). The kind of information you'll get from each of these functions is obvious from the name of the function, except for GetWindowFileModuleName( ), which returns the name of the EXE or DLL that created the window in the first place. Normally, you'll need to pass the HWND for the window that you want to ask about as part of the call. These functions will ask for a pointer to a structure as well, which will be filled with the required information on function return.

As you can see, the additions to the USER subsystem are substantial. This section has given you only a brief overview of the new features that Microsoft has added. To fully describe and demonstrate all of the features could easily take an entire book.

GDI

The Graphics Device Interface (GDI) subsystem is the part of Windows associated with printing and drawing on screen. There aren't as many changes to this system, but the few that are present will make a difference in your programming efforts.

As with the USER subsystem, one of the first areas that we'll look at in the GDI subsystem are some new messages that Windows 2000 supports. In this case, there's only one new system message of any significance. The WM_SYNCPAINT message synchronizes painting so that independent GUI threads don't get linked. The GDI automatically determines whether it needs to send this message when a child window has been moved, hidden, shown, or resized. This message gets sent to the top-level windows of other threads so that repainting gets performed as needed by all threads displayed on screen.

Another set of GDI changes has to do with the window animation feature discussed in the previous section. It turns out that the animation actually relies on features found in the MSIMG32.DLL. Right now this DLL only contains a few functions, but they're important. For example, the AlphaBlend function will blend the rectangular area of one device into another device context. If the rectangles aren't the same size, then AlphaBlend can perform the stretching required to make the rectangle fit.

You'll need to use the WINGDI.H file that comes with the Platform SDK to use any of the new MSIMG32.DLL features. Fortunately, these features don't require you to download a special SDK as some of the other features listed in this chapter do.

Also included with MSIMG32.DLL is a new function for performing fills called GradientFill( ). This is the function that's responsible for the new look in the Windows title bars. It allows you to fill an object with a gradient of colors that moves from one color to another.

NTFS 5

We've already covered a couple of the interesting changes to NTFS: FAT32 support and UDF in the introduction to the chapter. However, there are a lot of other interesting changes you should know about that affect NTFS 5 directly. For one thing, NTFS 5 supports reparse points. Essentially, this is a change in the way that NTFS handles file paths. Each step of the directory path is individually parsed and, if necessary, redirected to another location. For example, if you had a path of C:\Windows\README.TXT, the path would get parsed once for C: and again for Windows. This capability will eventually allow for multiple drive links into one drive. You could connect to any drive on your machine using a single drive letter. This feature is also extensible. You can create your own reparse filters that adjust the way that the file path gets parsed.

If you decide to build your reparse filter, you'll need the Installable File System (IFS) kit. The IFS kit is available at `http://www.microsoft.com/hwdev/ntifskit/`. This is also the site where you apply for a reparse point tag—the mechanism used to identify your reparse filter. The only time when you'd really require a reparse point tag is if you intended to make your reparse filter available for public use. You can also sign up for a mailing list at this Web site that will keep you up to date on the latest IFS developments.

Another feature is per-disk and per-user quotas. The use of quotas will keep one person from consuming all of the space on a server hard drive. Of course, this applies equally well to processes or any other object as well. Before the use of quotas, an errant application could fill an entire hard drive with something like log file entries and the administrator might not realize there was a problem before it was too late to do something it. Quotas are based on the Security Identifier (SID) of the owner—whoever created the file is responsible for the space that it consumes.

Now is the time to switch to the new GetDiskFreeSpaceEx( ) function in place of the older GetDiskFreeSpace( ) function. Either function will work well with quotas, but you won't get support for several new Windows 2000 features using GetDiskFreeSpace( ). For one thing, GetDiskFreeSpaceEx( ) returns two values: The true amount of free space left on the hard drive and the amount of space that's available to the current user. The GetDiskFreeSpaceEx( ) function also maps down to the physical disk associated with a DFS tree, which means that you get true free disk space, rather than the space that's available at the root of the DFS tree itself. GetFreeDiskSpaceEx( ) is designed to account for future changes in the quota system. For example, you may find that in the future quotas are assigned directory level rather than the volume level. Finally, GetFreeDiskSpaceEx( ) will report the true size of volumes larger than 2GB, a problem with the older function.

NTFS 5 also supports native property sets. This feature is like the property pages that you attach to a COM object. They allow you to define attributes for a file. For example, you could attach a description of a graphics image to a graphic file using this technique. You can create your own custom property sets using the IPropertySet and IPropertySetStorage interfaces. One of the perks of using this new feature is that Microsoft Index server is set up to use property sets. Anything you attach to a file will be used to index it, which makes searches much easier for the user.

Tracking all of the changes that occur to the file system is another NTFS 5 feature. You can record things like file creation, deletion, and modification, as well as changes to properties, security, encryption, and compression. In short, you can now track how the file system gets used.

You'll really like the new sparse file support in Windows 2000. When creating a normal file, the file size reflects every byte that you actually use. A sparse file, on the other hand, allocates a certain number of bytes, but doesn't actually have to use them. This is termed as allocation on demand. It allows you to create a file

that'll provide enough space for what you need to eventually store, rather than allocate bytes as you use them. Fortunately, the system only "charges" you for the bytes that are actually in use. For example, you can allocate a 1GB file. When the user clicks on that file, they'll see a 1GB file. However, if you've only placed 64KB of data in that file, then the quota system will see that you've only used 64KB of your allocated space on the drive.

DirectX 6.0

Both Windows 98 and Windows 2000 will include DirectX 6.0 right out of the box. Of course, users of older versions of Windows can download DirectX 6.0 and install it as well. In fact, you may find DirectX 6.0 installed automatically with certain types of Microsoft products and a large number of games. Interestingly enough, you'll find DirectX 6.0 used with such Microsoft products as Internet Explorer.

The three words that describe the DirectX 6.0 update the best are processors, display hardware, and 3D processing. This latest version of DirectX is designed to make the maximum use of processor features like Intel's MMX and AMD's 3DNow! instruction sets. This means that the applications you write will gain a lot of advantages in the way of performance.

Microsoft has also been working with the various hardware vendors to ensure that DirectX 6.0 supports all of the latest features in display hardware. Direct3D has been fully optimized to make use of whatever 3D hardware the user has available on their machine. (DirectSound has also received minor optimization improvements, though not to the same degree as Direct3D.) If the user doesn't have any 3D hardware, or the hardware is missing some features, then Direct3D will make up for the lack using software emulation. Obviously, there isn't any way to optimize the software emulation enough so that it can compete with a hardware solution, but Microsoft has gone a long way toward closing the gap (unfortunately, they have a much longer way to go).

You'll find a lot of new 3D features in DirectX 6.0. For example, this version emphasizes the ability to work with multiple textures and to place those textures on a single object to give it life. You'll also find advanced hardware features like bump mapping and compressed texture mapping. Bump mapping is used to add more surface detail to an object, while compressed texture mapping makes the texture map smaller so that it requires less memory to store. Some of these new technologies weren't created by Microsoft; they were licensed from other companies and added to DirectX 6.0 to give it fuller functionality.

DirectX 6.0 requires the use of a special SDK if you want to add any of the new features to your application. You can download a copy of the DirectX 6.0 SDK at `http://www.microsoft.com/directx/download.asp`. Unfortunately, this SDK weighs in at a whopping 80MB and there isn't any way to download sections of the SDK as there is with the Platform SDK. As a result, it would probably be better to order the CD containing the SDK, which is available at the same Web site.

Adding all of this new functionality to Direct3D requires new and updated interfaces. The following list will give you some idea of what has changed since DirectX 5.0.

- IDirect3DRM3
- IDirect3DRMAnimation2
- IDirect3DRMAnimationArray
- IDirect3DRMAnimationSet2
- IDirect3DRMClippedVisual
- IDirect3DRMDevice3
- IDirect3DRMFrame3
- IDirect3DRMMaterial2
- IDirect3DRMMeshBuilder3
- IDirect3DRMObject2
- IDirect3DRMShadow2
- IDirect3DRMTexture3
- IDirect3DRMViewport2

One of the things you won't find in DirectX 6.0 on other machines, but will find on Windows 2000 is DirectMusic. While DirectSound works with digital sound files, DirectMusic will work with message-based musical data. DirectMusic will convert these messages to a wave-file format for presentation on the PC. The default message-based file format for DirectMusic is the one supported by the Downloadable Sounds (DLS) standard. DirectMusic also supports Musical Instrument Device Interface (MIDI). As with all COM technologies, DirectMusic will rely on a series of interfaces to work its magic, including IDirectMusicCollection, IDirectMusicObject, and IDirectMusicDownloadedInstrument.

MAPI (Messaging API)

About the only news about MAPI is no news at all. As of this writing, it appears that the level of MAPI support you got in Windows NT 4.0 is the same level of support that you'll get for MAPI in Windows 2000. This isn't all bad. Considering how many other features have changed, it's nice to know that at least some part of the operating system is stable.

TAPI (Telephony API)

The latest news for TAPI and Windows 2000 is that you'll get your first glimpse of TAPI 3.0 in this version of the operating system. The main technological leap that Microsoft is touting for this version of TAPI is that it combines IP telephony and traditional Public Switched Telephone Network (PSTN) telephony. In short, you can have a conversation with someone somewhere and not really care how the connection was made. At least, that's the theory behind what will happen.

Not only will combining IP and PSTN telephony make things easier for the user, it'll save organizations money as well. Right now an organization has to pay two completely different groups to manage telephony. Obviously, reducing this to one cohesive group that can manage not only voice, but also data transfers, over any media is a big advantage.

From a programmer's perspective, there are several interesting features in TAPI 3.0. The most important feature is that unlike TAPI 2.1, TAPI 3.0 is implemented as a series of COM objects. This means that you'll be able to use standard COM programming techniques with TAPI 3.0, which requires a knowledge of the interfaces needed to interact with the API. In addition to the COM portion of TAPI 3.0, there are three other elements that you need to know about.

- **TAPI Server:** Abstracts the TAPI Service Provider Interface (TSPI) so that TAPI 3.0 and TAPI 2.1 service providers will both work as before.
- **Telephony Service Providers (TSPs):** The service provider translates the protocol independent calls used by the main portion of TAPI and translates them into protocol-specific calls.
- **Media Stream Providers:** A special part of TAPI that handles DirectShow media streaming needs. (Generic streams are handled by the application.)

As previously mentioned, TAPI 3.0 relies on COM objects instead of the function calls used by the previous version. Obviously, this makes it easier for you, as the application developer, to extend the native capabilities of TAPI to meet your personal needs. There are five objects that you'll interact with when using TAPI. The following list describes them for you:

- **TAPI:** You'll use this as your point of entry into TAPI. It allows you to see all of the telephony resources available to the local computer and enumerate them prior to use. The process of enumerating the resources provides you with a list of local and remote addresses that you can use to create an Address object.

- **Address:** The Address object provides a link from the local to the remote location. You need a local address (the call origination point) and a remote address (the call destination) to complete a call. Once the call is completed, TAPI creates a Call object. One of the ways that you can use the Address object is to find out about the media used to transfer information.
- **Terminal:** This is the termination for the point of origination for a call. This is the object that will handle input/output. Normally, it would be a handset or a microphone/speaker combination. However, you can create terminal objects that consist of files or other standard objects as well. The only criteria are that the Terminal object handle the flow of information.
- **Call:** Imagine this object as the first-person view of the connection between a local and remote address. There's one Call object for each entry in the Call-Hub object. In other words, each remote location is handled individually.
- **Call-Hub:** A representation of all of the participants in a call. You can use this object to enumerate the participants and find out more about them. The Call-Hub object is created automatically for you by TAPI—it can't be created by an application.

Not only does TAPI handle your calls, it's also integrated with Active Directory. There are two benefits to allowing integration. First, Active Directory maintains a directory of remote addresses that you can call—at least those within the company. Second, since the IP address for each workstation will change on a daily basis, you can't hard code it into an address book. However, Active Directory notes the IP address of each remote address as the workstation logs into the network, so getting a current IP address from Active Directory is relatively easy. The actual level of integration between Active Directory and TAPI is much more complex than what I've explained here, but this provides you with a good starting point to learn more.

SAPI (Speech API)

SAPI provides the means to synthesize and recognize speech on a computer. Unlike many of the technologies that we've looked at, you can use the same version of SAPI on any 32-bit Windows platform. The same version you use with Windows 2000 will work with Windows 95 or Windows NT 4.0.

The SAPI SDK isn't included as part of the Platform SDK or Visual C++ 6.0. If you plan to implement some form of speech processing in your application, then you'll need the SAPI SDK. There are two versions of this SDK. The first contains some speech recognition and synthesis engines (39MB), while the second contains just the files you need for application programming purposes (7.9MB). The SAPI SDK is available at `http://www.microsoft.com/IIT/download.htm`.

As with many new Microsoft technologies, SAPI relies on a series of COM interfaces to perform its work. The current version of SAPI is 4.0a, which is the version you'll use with Windows 2000. The following table tells you which interfaces have changed since the previous version of SAPI.

Table 2-2
Changes in Interfaces

Interface	*Description*	*Latest Version Changes*
ISRSpeaker2	This interface works with the speaker information. It enables you to perform a variety of tasks like train the computer or move the current speaker's profile to another computer so that you won't have to train it.	The latest version of this interface adds functions to commit the speaker's profile to disk, gain information about the current speaker, and rename the speaker training data file.
IVDctCommandsBuiltIn and IVDctCommandsApp	Microsoft has divided the IVDctCommands interface into these two new interfaces. The original purpose of this interface was to control the list of commands available to the speaker during dictation.	The latest version of the interface allows you to store these commands to disk (or other locations), activate/deactivate commands, or work with the speech engine grammar.
IVDctGUI2	Use this interface to control the correction window used during dictation to change misinterpreted entries.	The latest version of the interface provides the capability to tell Voice Dictation to use the GUI's engine for dictation or insertion. There are also functions to change the size of the rectangle used to display the correction window.
IVDctNotifySink2	Essentially this interface acts as a sink for all of the status information generated during dictation. For example, this is the interface that receives voice-level information.	The latest version of the interface provides functions that return errors and warning information.
IVDctPlayback	This is an entirely new interface that allows the user to play back information recorded during a dictation session. There are methods to start and stop the playback, along with one method that returns the current playback status (playing or stopped).	New for this version.
IVDctText2	All of the input from a dictation session goes into a virtual textbox for processing. This interface allows you to control the contents of the virtual textbox. In addition, you can use these methods to create bookmarks and lock a current set of words.	All that the latest version of the interface adds is more text manipulation features.

Continued

Table 2-2 *(continued)*

Interface	*Description*	*Latest Version Changes*
IVDctTextCache	Every voice dictation object maintains bins where it stores text and results objects. The functions in this interface allow you to copy and paste elements contained in the various bins.	New for this version.
IVoiceDictation2	Allows an application to register with a voice dictation object and to control the recognition of speech.	The latest version of the interface includes a function that allows the application to mimic a command being spoken into the current site.

HID (Human Interface Device)

HID is part of DirectInput and a new part of Windows 2000. The main purpose for this set of interfaces is to decouple the hardware from applications. It's designed to support human interface devices like keyboards, mice, and joysticks. HID will also support a limited number of control devices (although Microsoft was unclear on exactly what control devices it would support).

In addition to decoupling the hardware from applications, HID will allow devices to provide some level of self-description. In other words, the device will provide some type of input about its type and characteristics. An application could then select the information it needed from a database of facts about the device.

There are some differences between Windows 98 and Windows 2000 when it comes to HID support. For one thing, Windows 98 will continue to provide support for legacy game ports that require VJOYD.VXD and VxD mini-drivers. It'll also provide support for DOS games that require a game port using game-port emulation. Windows 2000, on the other hand, will only provide game-port support using HID emulation and HID mini-drivers. There will be no support for DOS games at all in Windows 2000.

From early HID documentation, it appears that each HID driver will also support a COM-class interface, but Microsoft hasn't spelled out exactly how this support will work. There are three major components required to implement HID: client, class driver, and mini-class driver. The HID client interacts with the application program. The HID class driver provides core service support, while the mini-class driver provides hardware-specific support.

ACS (Admission Control Service)

At the most basic level, ACS is a method for limiting the amount of network bandwidth that any one individual can use based on a policy stored in Active Directory. This will prevent high-bandwidth applications like video from reducing network throughput to a crawl. It'll also enable the network administrator to reserve network bandwidth for high priority applications like IP telephony. This all boils down to maintaining quality of service (QoS) for the user.

The inner workings of ACS are relatively easy to understand. An application or user will request a specific amount of network resources (bandwidth for the most part) using the ReSerVation Protocol (RSVP). RSVP sends a message to ACS, which compares the network resources requested by the application's user against those stored in Active Directory. If the user has sufficient resources to honor the RSVP request, then ACS returns the request approved. Otherwise, the request is denied and the whole process stops until the user can get sufficient rights to allow the RSVP request to succeed.

Interestingly enough, there isn't an SDK available for ACS as of this writing, but there probably will be by the time that you read this chapter. The best place to look for additional information about ACS is `http://www.microsoft.com/HWDev/network/cable/qos-cable.htm`. This site provides the whole in-depth story of how Microsoft plans to implement QoS within its various products. This is also the place where you'll find links to other Web sites containing additional information—making it the most likely place to find out more information about a future SDK. Another good place to look for information about ACS is `http://www.microsoft.com/hwdev/devdes/qos.htm`. This site contains additional information about both ACS and QoS. It also contains a pointer to the DDK where these new features are documented (albeit at a very low level).

Part of the ACS decision is based on current network conditions. The current IEEE 802.1p specification allows for up to seven-packet priority levels. Even if a user is normally allowed to reserve network bandwidth for data transfer, the request could be denied if a higher priority needed the bandwidth first. Part of the ACS administration process then is to assign priority levels to packets so that high-priority tasks get processed first. As you can see, the combination of priority levels and user resource limitations will make QoS easier to guarantee on a Windows 2000 network.

Security Services

Previous sections of the chapter have covered new Windows 2000 security technologies like Kerberos (you can be sure we'll cover these features in a lot more detail later in the book as well). This section looks at the programming part of the picture, the security services. As with every other part of Windows 2000, Microsoft has added a lot of new security features. However, in this case, you can easily categorize the features by API. Table 2-3 contains a list of the new or updated APIs and what they'll do for you.

Table 2-3
Windows 2000 New or Improved Security APIs

API Group	*Description*
AccessCheck	Determines whether a security descriptor grants a user a specific set of access rights. All of the functions check by type, but you can also choose either or both, result list or audio alarm. These functions includeAccessCheckByType(), AccessCheckByTypeAndAuditAlarm(), AccessCheckByTypeResultList(), and AccessCheckByTypeResultListAndAuditAlarm().

API Group	*Description*
AddAccess	Allows you to add an access allowed or access denied Access Control Entry (ACE) to the end of a Discretionary Access Control List (DACL). The object form of these functions will allow the ACE to audit access to an object, a property set, or an object property. These functions include AddAccessAllowedAceEx(), AddAccess AllowedObjectAce(), AddAccessDeniedAceEx(), and AddAccess DeniedObjectAce().
AddAudit	Allows you to add a system audit ACE to the end of the System Access Control List (SACL). The object form of these functions will allow the ACE to audit access to an object, a property set, or an object property. These functions include AddAuditAccessAceEx() and AddAuditAccessObjectAce().
Convert	Convert an Access Control List (ACL) to a security descriptor or vice versa. This group contains three functions, one for each type of conversion you can perform: ConvertAccessToSecurityDescriptor(), ConvertSecurityDescriptorToAccess(), and ConvertSecurityDescriptor ToAccessNamed().
SecurityInfoEx	Updates of the current set of SecurityInfo functions. The Ex versions of the functions allow you to get information about the security of object properties in addition to the objects themselves. These functions include GetNamedSecurityInfoEx(), GetSecurityInfoEx(), SetSecurityInfoEx(), and SetNamedSecurityInfoEx().
SetEntriesIn	Creates a new ACL by merging new access control information into an existing ACL. There are two functions, one for access lists (DACL) and another for audit lists (SACL). These functions include SetEntriesInAccessList() and SetEntriesInAuditList().
PrivateObjectSecurity	Extensions of existing functions that allow you to create or modify security descriptors. The Ex versions allow you to control how a child of the parent container inherits the security descriptor. These functions include CreatePrivateObjectSecurityEx() and SetPrivate ObjectSecurityEx().

There are also functions that allow you to perform encryption and description. The two new functions, EncryptFile( ) and DecryptFile( ), are part of ADVAPI32.DLL. These two functions are implemented as a filter driver and only work with NTFS formatted drives, and not with any file in general. From the user's perspective, the file hasn't changed at all when it's encrypted. However, unauthorized access is quickly discouraged. The only problem with this particular feature is that an encrypted file can't be compressed, and vice versa.

Summary

This overview should have shown you one thing: Microsoft had a hard time meeting deadlines for Windows 2000 because the operating system had to go through a major upgrade at every level. Windows 2000 isn't limited to just a few new APIs or functions; you're seeing a fundamental change in the way that Microsoft plans on handling many operating system features. With this in mind, you can only imagine what will take place in the future as Windows 2000 grows. Here are some additional conclusions you can draw from the contents of this chapter:

- Windows 2000 is a vast improvement over previous versions. About the only problem that you might encounter is figuring out how to make use of these new features. As usual, the amount of new documentation has kept pace with the size of the rest of the operating system.
- The USER subsystem includes a lot of new input options, many of which appear to center around the mouse. Some of the more interesting additions are the functions that are designed to make using a mouse with a menu easier.
- Microsoft has worked hard to make more information available to the programmer. For example, there are a variety of functions like GetTitleBarInfo( ) that enable you to obtain information about previously inaccessible objects.
- When looking for new GDI subsystem features, it really pays to look at the contents of the MSIMG32.DLL file. There are new features like AlphaBlend that allow you to perform advanced graphics manipulation on standard Windows objects.
- NTFS 5 offers a wealth of new features that users will eventually notice, such as FAT32 support and UDF, which is required to support DVD drives.
- There are also NTFS 5 features that are definitely programmer-oriented such as the ability to use reparse points and the new features of GetDiskFreeSpaceEx( ).
- DirectX 6.0 is designed to make maximum use of the user's hardware and processor features to speed complex operations along. Most of these optimization efforts center on the latest Direct3D updates.
- DirectMusic, which wasn't shipped as part of DirectX 6.0, provides the ability to convert message-based input into wave output. It works with both DLS and MIDI.
- The major news about TAPI 3.0 is that it uses COM objects in place of the older programming interface found in TAPI 2.1. This will make writing applications for TAPI a lot easier and allow a programmer to extend TAPI as needed to meet application needs.

- SAPI 4.0a provides a lot of new features that make it easier to train the speech system to recognize a speaker's voice, reduce the work required to allow a speaker to move from one machine to the next, and improve handling of dictated information.
- The main reason to use HID is to decouple the application program from the underlying hardware. This allows the hardware and software to work independently of each other. In addition, it's more efficient because the hardware can self-describe itself one time, then the software can rely on entries in a local database to gain hardware-related information as needed.
- ACS is the major component that guarantees network QoS on a Windows 2000 network. There are two major considerations when granting user RSVP requests for network bandwidth: the user's assigned bandwidth limit and the priority of the packets that the user wishes to transmit. The user's bandwidth limit is stored in Active Directory, which means that the information is available network-wide.
- Windows 2000 has a lot of new security features. Most of the new features are designed to enhance object security and permit access to object properties. However, there are other function updates that'll make it easier to add security to an application in general.
- Windows 2000 provides support for file encryption and decryption through the EncryptFile( ) and DecryptFile( ) functions found in ADVAPI32.DLL. These new functions only work on NTFS-formatted drives, and you can't use them on compressed files.

✦ ✦ ✦